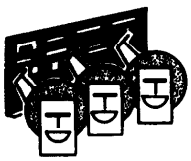
Vol. 33 No. 6　　情　報　処　理　　June 1992

リレー解説



海外の並列処理研究動向

MIT における並列処理研究の現状†

坂　井　修　一††

1. はじめに

筆者は，平成3年4月より1年間，米国マサチューセッツ工科大学（MIT）に長期出張した．MIT では Laboratory for Computer Science (LCS) に籍を置き，並列計算機アーキテクチャの研究に従事した．

本稿では，MIT—LCS における並列処理研究の現状について，実際にシステムを構築している三つの研究を中心に解説する☆．

2. データフロー／マルチスレッド計算機

MIT はデータフロー計算機の発祥の地[1]である．データフローは，計算をデータ依存関係に基づく有向グラフとして表現し，これを効率よく処理する並列計算機アーキテクチャとプログラム言語を開発することを目標とする概念（あえていえば「思想」）である．

Dennis 教授（現在 MIT 名誉教授）の最初の提案[1]ののち，米，英，日，加，蘭など各国で精力的にデータフローの研究がすすめられており，MIT も依然として研究の中心の一つである．

アーキテクチャ的には，Arvind 教授の提案した Tagged Token Dataflow Architecture (TTDA)[2]が 1980 年代初期のエポックであり，カラーをもつトークンが要素プロセッサ (PE) を起動し，処理の結果生じたトークンが別の PE を起動する，という流れで並列計算が進行するアーキテクチャが提案された．ほぼ同時期に開発された英国マンチェスタ大学のデータフローマシン[3]と TTDA によって，循環パイプライン，トークンのマッチング，I 構造などの基礎的な機構が確立された．

1980 年代，MIT はアーキテクチャ技術の発展，とくに実機の開発において大きな進歩を遂げることができず，TTDA を基礎とする並列計算機の研究開発は日本の研究機関[4),5)]に遅れをとることになった．その後，Gregory Papadopoulos 助教授を中心として，実機 Monsoon[6] (図-1) の開発に成功した（1991 年）．後にも述べるように Monsoon はアーキテクチャ的にはすでに前世代のものであり，実装も8並列と小規模で，最高性能 80MFLOPS（倍精度）と，試作された時期を考えると，実性能にやや難があるものの，通信・同期・命令実行というパイプラインの連続性において魅力的な計算機となっている．

図-2 に Monsoon の PE アーキテクチャを示す．本 PE は8段の循環パイプラインをもち，トークンのマッチング機構として，連想機構をもたない ETS (Explicit Token Store) という方式を採用している．ETS は当該命令が陽に待ち合わせ番地を指定する方式で，マッチングに要する時間が大きくなる反面，ハードウェアが簡単で柔軟性が高

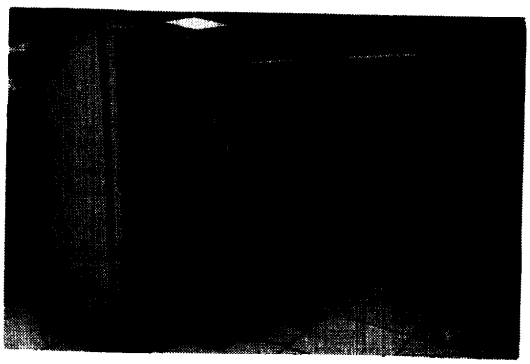


図-1　データフローマシン Monsoon（右は Arvind 教授，左は著者）

† Current Status of Parallel Processing Researches in Massachusetts Institute of Technology by Shuichi SAKAI (Computer Science Division, Electrotechnical Laboratory).

†† 電子技術総合研究所情報アーキテクチャ部

☆ 以下の本文中では，full professor を教授，associate professor を準教授，assistant professor を助教授と訳した．通常後二者は，それぞれ助教授，常勤講師と訳すが，若手 faculty の独立性の高い MIT では，上の訳のほうが実際のイメージに合うと考えたからである．

いという利点がある．Monsoonで最も問題とされることは，循環パイプラインだけの構成のため，局所計算・直列計算に弱く，高い処理効率は望めない，ということである．

Monsoonでもう一つ注目されるのは，その相互結合網である[7]．Monsoonの結合網は4×4のルータを構成要素とする蓄積交換の多段スイッチ網であるが，各ルータに局所的なフロー制御機構を導入することにより，閉塞率のきわめて低いパケット通信を実現している．各ルータは独自に開発したECLチップであり，ポートあたり100 MB/sの高い転送レートをもつ．

さて，1980年代後半より純粋なデータフローアーキテクチャへの批判が内部・外部より高まった．その結果，真に効率的な並列アーキテクチャは，局所計算をvon Neumann型のパイプラインで処理し，PE間通信をともなう処理だけをデータフロー的に処理するハイブリッド・アーキテクチャになる，という考えが主流となった．この考え方は，逐次実行スレッドをデータフロー的に駆動しフォンノイマン的に処理するマルチスレッド計算機の概念として再整理されてきている[8),9)]．

筆者の所属する電総研のグループは，データフローの内部に局所処理ブロックを設定し，パケット処理は循環パイプライン，局所処理はRISCのパイプラインで処理する新方式の計算モデル（強連結枝モデル）を提案し，本モデルに基づくプロセッサEMC—R[10)]および高並列計算機システムEM—4[11)]を設計・試作した．EM—4はPE数1024台のシステムであり，その最初のプロトタイプ（80 PEシステム）は1990年4月より最高性能1 GIPSで稼働中である．

MITはMonsoonの改良とEM—4で開発された新技術の摂取・発展を目的として，次世代の超並列計算機である＊T[12)]の研究・開発を始めた．＊TのPE（図-3）は，同期処理部とデータ処理部に大きく分かれ，前者が相互結合網インタフェースと待ち合わせを司り，後者がRISC型のデータ処理を司る．本PEは，モトローラ社のMC 88110を核とした実装になる予定で，1992年中にPE全体が完成する予定となっている．

以上がアーキテクチャ研究および実験機試作の概観であるが，現在のMITのデータフローグループの主力は，アーキテクチャ研究よりもむしろソフトウェアの開発にある．特に関数型言語Idおよびその処理系の開発[13),14)]には，常時4, 5人のスタッフおよび多くの大学院生が従事しており，最適化コンパイラの開発（ターゲットはMonsoon），Lawrence Livermore研究所などと共同しての実用的なプログラムの開発，Id in Idと

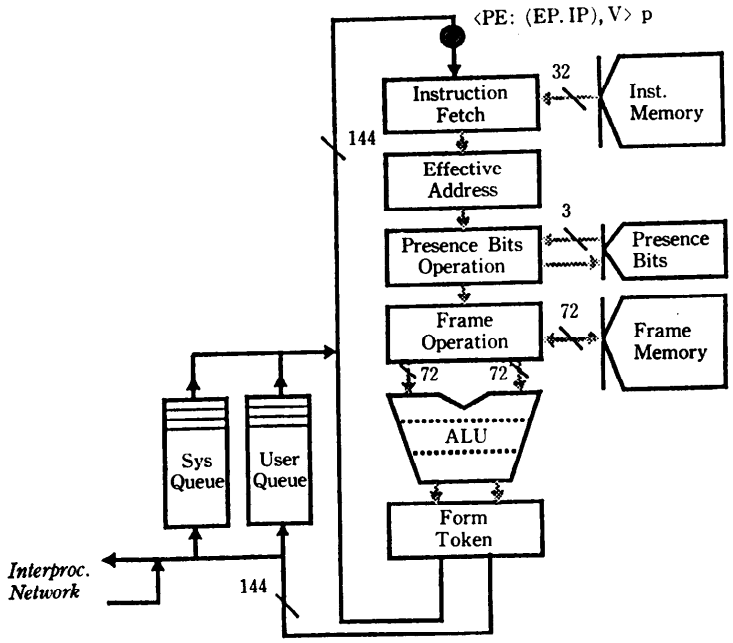


図-2 Monsoonの PE アーキテクチャ

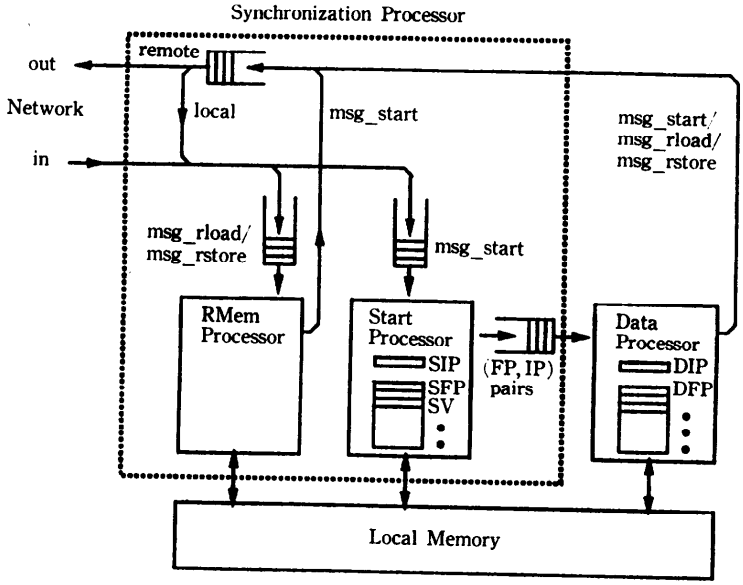


図-3　*T の PE 内部構成

呼ばれるセルフコンパイラの開発，同期構造を含むデータ型の研究，自動型チェック機構の研究などは注目に値する．また，Id のプログラム環境はよく整えられており，emacs の中からコンパイラや並列処理シミュレータを呼ぶことができ，並列度の時間変化をグラフにしてX端末に表示することなども自動的にできる．筆者もいくつかの応用問題を解いてみたが，Id のスタイルに慣れれば，エレガントなプログラムを書ける場合もあり，興味深いものであった（図-4）．

```
def point x y = (x, y);
typeof make_triangle = (F, F) -> (F, F) -> (F, F) ->
    (vector (F, F));
def make_triangle p1 p2 p3 =
    {vector (1, 3) of
     | [1] = p1
     | [2] = p2
     | [3] = p3};
typeof make_rectangle = (F, F) -> (F, F) -> (F, F) -> (F, F) ->
    (vector (F, F));
def make_rectangle p1 p2 p3 p4 =
    {vector (1, 4) of
     | [1] = p1
     | [2] = p2
     | [3] = p3
     | [4] = p4};
typeof translate = (F, F) -> (F, F) -> (F, F);
def translate (dx, dy) (x, y) = x+dx, y+dy;
typeof skale = (F, F) -> (F, F) -> (F, F);
def skale (sx, sy) (x, y) = sx * x, sy * y;
typeof rotate = F -> (F, F) -> (F, F);
def rotate theta (x, y) = { costh = cos theta;
                            sinth = sin theta
                            in ( x * costh - y * sinth,
                                 x * sinth + y * costh)};
```

図-4　Id のプログラム例
（三角形，四角形の表現と図形の平行移動，拡大・縮小，回転）

3. J-Machine/M-Machine

データフローの研究が，ハードウェアもソフトウェアも単一の計算原理から統一的に扱おうという方法論によっているのに対し，William J. Dally 準教授のグループでは，複数の並列計算モデルをいくつかのハードウェア・プリミティブで支援する超並列アーキテクチャと並列ソフトウェアシステムの研究開発をすすめている．

このグループがターゲットとする計算モデルは，共有メモリモデル（共有メモリアーキテクチャではない），データフローモデル，アクタモデル，データパラレルモデルなどである．Dally 準教授はこれらのモデルにもとづく並列計算は，通信，同期，ネーミングといったハードウェアの基本機構によって実現されること，したがって，特定のモデルに拘泥することない幅広い用途の並列計算機システムがこれらの機構をうまく組み合わせることによって実現できることを主張している．

Dally 準教授のグループで開発中の並列計算機は，J-Machine[15),16)] と呼ばれている（図-5）．

J-Machine は，メッセージ交換にもとづく計算

機である．メッセージは相互結合網から当該 PE に入ってくると，いったんキューに入れられ，PE が処理待ちの場合ないし優先度の低い処理を行っている場合には，すぐにこれが取り出されて処理される．J-Machine は2レベルの優先度処理を行うが，各 PE には優先度に応じてレジスタセットが用意されており，文脈の切り替え (context switch) が高速に実現される．また，行き先の論理番地（より一般的にはオブジェクト名などのキーワード）から物理番地への変換を支援する連想機構と専用命令が用意されていて，高速の並列化仮想記憶を実現している．さらに，各データ語上で同期をとるための future tag, c-future tag, 出力メッセージ用 FIFO バッファなど，要素技術に工夫がみられる．各 PE の性能は 10 MIPS である．

J-Machine は，通信網として3次元メッシュを採用し，最大 65,536 台の PE アドレッシングが可能である．通信ノード（スイッチ図-5 の 3D Router）は PE チップ上に実装され，局所的なフロー制御を行っている（ストアアンドフォワードデッドロックはここで回避されている）．通信の平均遅延は 800 ns であり，遠隔データにアクセスするさいのレーテンシは 2 μs 以下におさえられている．また，ポートあたりのスループットは 360 Mb/s となっている．

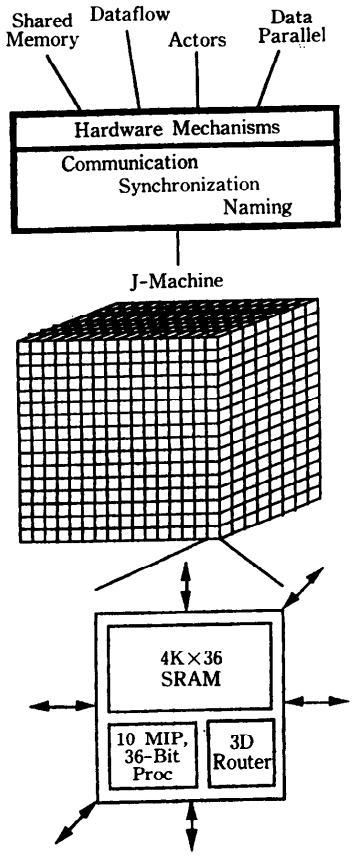


図-5 J-Machine の構成

J-Machine の研究で注目されるのは，近未来の実装技術（とくに VLSI 技術）への見通しをもってアーキテクチャを決めている点であり，「超並列計算機のアーキテクチャは，利用可能なシリコンの総面積をいかに有効に使うかでその価値が決まる」という考えを推し進めていることである．

1992年1月現在，J-Machine は，プロトタイプとして 1,024 台の PE からなるシステムを構築中であり，128 PE が 32 MHz のクロックで稼働中である．J-Machine の PE は，約 110 万トランジスタからなるカスタム LSI (intel 社が実装，144 Kbit のオンチップ SRAM を含む) 一個と DRAM からなり，60 cm×60 cm のプリント基板に 64 PE が実装されている．1,024 PE システムは本基板 16 枚よりなり，実装規模としてはきわめて小さなものとなる予定である．

ソフトウェアとしては，Concurrent Smalltalk, 並列 C, Id のコンパイラを実装し，また J-Machine を安全かつ効率的に動かすためのランタイムシステムを開発した．このうち，Concurrent Smalltalk では物理的に分散配置されたオブジェクトが実現されている．

さらに Dally 準教授は，J-Machine に続いて，より野心的な M-Machine[17] の研究開発に着手している．

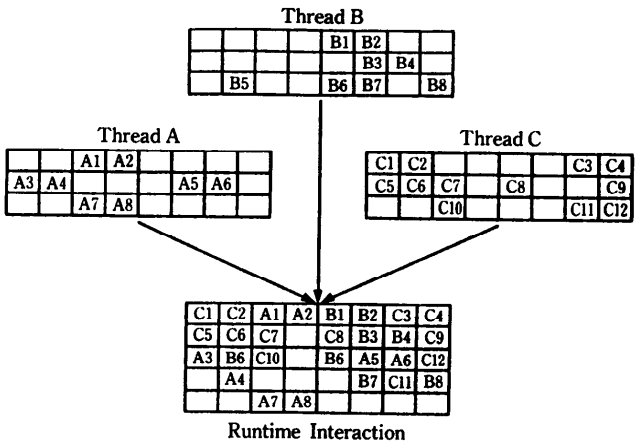
M-Machine は，1995 年の実装技術を想定した超並列計算機である．M-Machine を構成するチップは 0.5 ミクロンルールの ULSI を想定し，ここに4台の Operational Unit (OU) と 8 Mb の SRAM を載せることを考えている．各 OU は，整数演算ユニット，浮動小数点ユニット，レジスタ，キャッシュなどをもつスーパスカラ計算機である*．

M-Machine で特徴的なことの一つは，複数

* 本稿の執筆段階では，M-Machine は構想およびシミュレーションの段階であり，PE の構成図や目標性能など，詳しい資料は公開されていなかったため，本文の記述も，PE チップについてのさわりとスケジューリング原理の説明だけにとどめておく．

のスレッド（連続した処理をほどこすプログラムの部分）をインタリーブして各OU に割り付ける最適化を行うことである（図-6）．一つのスレッドでも内部に並列性のある場合は，OU 内あるいはOU 間でこれを並列処理するのだが，その場合はコンパイラが静的にOU の割り付けを行う．さらに複数のスレッドを割り付ける場合，1チップ上の全ての OU が順番に一つずつスレッドを処理するのではなく，割り付け可能な OU には．他のOU が処理しているのとは異なるスレッドを同時に割り付けるスケジューリングをする．本スケジューリングはコンパイラによる静的手法では実現できないため，実行時に動的に行われる．具体的には，OU 内のスコアボーディング機構がスレッドのインタリーブを支援している．

以上のスケジューリング手法は，Processor Coupling と呼ばれ，細粒度並列計算のスケジューリングで最も新しい話題の一つである．



スレッド A，B，C はそれぞれコンパイラによって静的に8つのOUに割り当てられる．実行時にはこれら三つのスレッドは動的にインタリーブされて処理される．

図-6 Processor Coupling

4. Alewife

Alewife[18] は，Anant Agarwal 助教授のグループで研究・開発中の分散型共有メモリ方式の並列計算機である．

近年活発に研究が行われたスヌープキャッシュ型の共有メモリによる並列アーキテクチャは，バスの容量の点から，PE 数の大きな超並列アーキテクチャには不向きであるとされている．データの配置が楽でプログラムが書きやすいという共有メモリ型の利点を失わずに，スケーラブルなアーキテクチャを構築するにはどうすればいいか，というのが Alewife の研究の動機である．

図-7 に Alewife の全体図を示す．Alewife の相互結合網は2次元格子網であり，通信はメッセージ交換方式で行われる．メモリは，論理的には全空間に一次元的なアドレスを割り振る共有型であ

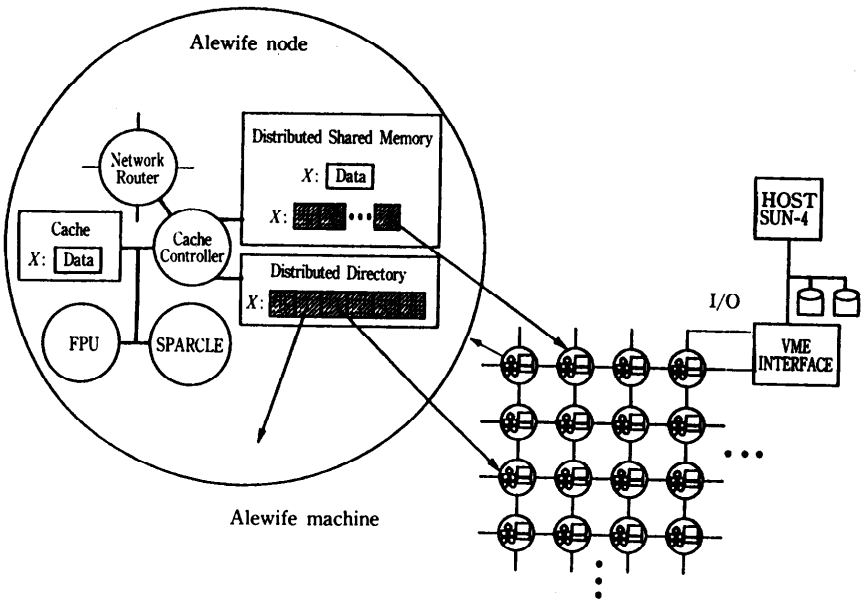


図-7 Alewife

り，物理的には各 PE に局所記憶の形で分散されている．PE にはコヒーレントキャッシュが置かれており，局所的なデータも，遠隔データもここにキャッシュされて用いられる．プロセッサ本体は，RISC 型のマイクロプロセッサである SPARC を改良した SPARCLE というチップを用いている．

Alewife は，並列計算機のネックとされるレーテンシの問題に対し，次に述べる二段構えのアプローチをとっている．

［1］ レーテンシの最小化

ネットワーク結合型のアーキテクチャで高い効率をあげることのできる新しい方式の並列コヒーレント・キャッシュを導入した．

Alewife のような分散型共有メモリ方式では，キャッシュによる参照関係を示すディレクトリをメモリに設けなくてはならないが，参照関係をいくつ保持し，どのようなプロトコルでこれを扱うかが，性能の決め手となる．Agarwal 助教授のグループでは，4 つのエントリをもつディレクトリを用い，これが溢れた場合にはトラップによってソフトウェア的にディレクトリを拡大する手法を採用している (LimitLESS 方式)[19]．Alewife の PE はトラップ処理が高速なため，本方式で，初めから完全なディレクトリをもつ方式（ハードウェア量の点で非現実的）に匹敵する性能が得られるという．

また，Alewife では，コンパイラによる問題の最適分割の研究，ランタイムシステムによる動的スケジューリングの研究などもなされており，高効率のキャッシュとあわせて，レーテンシの最小化をはかっている．

［2］ レーテンシの緩和

遠隔地データのアクセス時には，その到着を待つ間プロセッサをアイドルにすることは，性能の大幅な低下につながる．遠隔地データを待たなくては処理の進まないプロセスは，一度これを中断して実行環境を保存し，別のプロセスを当該 PE で実行することが必要である．Alewife では，このような遠隔地アクセスに起因する分脈の切り替えを高速にできるよう，(1)レジスタウィンドウの複数スレッドによる利用，(2)スレッド単位のスコアボードの導入，(3)複数スレッドに対応するような FPU の改良，(4)トラップ機構の強化，などをはかっている．商用の SPARC チップから SPARCLE への改良は主にこれらの点に関してなされている．

次に，Alewife のハードウェアの概略と開発の現状を述べる．

Alewife の PE は，上に述べたプロセッサ SPARCLE と，浮動小数点ユニット (FPU)，キャッシュ，キャッシュ・コントローラ，分散型共有メモリ，分散ディレクトリ（主メモリとは別バンク），相互結合網のルータからなり，一枚の PCB に実装される．SPARCLE は LSI Logic 社により 1992 年 2 月に，40〜50 MHz のクロックで実働化する予定である．キャッシュ・コントローラは MIT の制作，ルータは Caltech の Seitz 教授の開発したものを用いる．4 PE ないし 16 PE システムが今年 8 月に，64 PE システムは遅くても今年中に稼働する予定である．

ソフトウェアは，MUL-T という並列 LISP，Semi-C という並列Cを目下サポートしており，現在，最適化コンパイラを開発中である（いくつかの特定の問題についてはすでに開発に成功している）．

5. おわりに

本稿では MIT の並列処理の研究について，実際にハードウェアシステムを構築中の三つのグループの活動を紹介した．データフロー型，メッセージ交換型，共有メモリ型と種類の異なる三つの研究が一つの建物のなかで競いあっており，公的な場ではげしい論争の場面に出くわすこともある．また，LCS の廊下やラウンジで突然はじまるディスカッションも刺激的である．一つの大学内で三種類の（実装を含めた）大規模並列計算機の研究ができる環境は，日本の大学などではなかなか得られないものであり，羨ましく感じることもあった．

ここで述べた以外にも，問題に応じて PE の接続を自由に調節できる並列計算機の研究を行っている Steve Ward 教授，CM—5 の相互結合網を設計した Charles E. Leiserson 準教授，相互結合網や LSI 技術などの要素技術から超並列計算機のありかたを研究している Thomas F. Knight Jr. 博士などが顕著な活動をしている．

現在は，Multics の時代とは違って，「計算機の

研究は万事 MIT が中心」，ということはないが，MIT—LCS は世界の計算機研究の巨大な一拠点であり続けていることは確かであり，筆者の専門の並列計算機の研究でも質・量ともにトップクラスの充実度である．

最後に筆者の個人的な感想を述べて本稿を閉じたい．予算の問題を別にすれば，研究内容も研究者や大学院生の質も，筆者がこれまでに日本で見てきたものと MIT で見たものには大きな差がなかった．たとえば電総研の計算機環境，開発された並列計算機，人的資源など，どれをとっても MIT に劣るものはないと考える．ただし，新しい概念や体系を生み出す構想力と，それを定着させ，発展させる強い意志力は，見習うべきところがあると感じた．自戒をふくめて，これからの日本の研究者・研究機関に多く望まれるものだと思う．

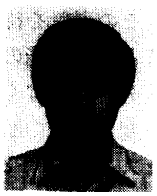
謝辞 本稿の執筆にあたって，ご協力をいただいた Arvind 教授，Gregory M. Papadopoulos 助教授および Andy Boughton 博士，快くインタビュに応じてくださった，William J. Dally 準教授，Anant Agarwal 助教授，Thomas F. Knight Jr. 博士に感謝いたします．

なお，Arvind 教授は今年7月より1年間，東大工学部の客員教授として日本に滞在される予定であり，本邦の研究者との活発な交流が期待されます．

参 考 文 献

1) Dennis, J. B.: First Version of Data Flow Procedure Language, Lecture Notes in Computer Science, 19 G., Springer-Verlag, New York (1974).
2) Arvind, Kathail, V. and Pingali, K.: A Dataflow Architecture with Tagged Tokens, Tech. Rep. TR-174, MIT-LCS (1980).
3) Gurd, J., Kirkham, C. C. and Watson, I.: The Manchester Prototype Dataflow Computer, CACM, Vol. 21, No. 1, pp. 34-52 (1985).
4) Amamiya, M., Takesue, M., Hasegawa, R. and Mikami, H.: Implementation and Evaluation of a List-Processing-Oriented Data Flow Machine, Proc. of 13 th ISCA, pp. 10-19 (1986).
5) Hiraki, K., Sekiguchi, S. and Shimada, T.: System Architecture of a Datalow Super-computer, TENCON 87, pp. 1044-1049 (1987).
6) Papadopoulos, G. M. and Culler, D. E.: Monsoon an Explicit Token-Store Architecture, Proc. of 17th ISCA, pp. 82-91 (1990).
7) Joerg, C. and Boughton, A.: The Monsoon Interconnection Network, Proc. of ICCD '91, pp. 156-159 (1991).
8) Stalker, K. T. (chair): Multithreaded Parallel Architecture Based on Dataflow Principles: The Road To Commercialization? Workshop in Supercomputing 91 (1991).
9) Gao, G. R. (chair): Multithreaded Computers, Workshop in Supercomputing 91 (1991).
10) Sakai, S., Yamaguchi, Y., Hiraki, K., Kodama, Y. and Yuba, T.: An Architecture of a Dataflow Single Chip Processor, Proc. of 16th ISCA, pp. 46-53 (1989).
11) Sakai, S., Kodama, Y. and Yamaguchi, Y.: Prototype Implementation of a Highly Parallel Dataflow Machine EM-4, Proc. of 5th IPPS, pp. 278-286 (1989).
12) Nikhil, R. S., Papadopoulos, G. M. and Arvind: *T: A Multithreaded Massively Parallel Architecture. to appear in 19th ISCA (1992).
13) Nikhil, R. S. and Arvind: Id: A Language with Implicit Parallelism, CSG-Memo 305 MIT-LCS (1990).
14) Nikhil, R. S.: ID Language Reference Manual, Ver. 90. 1, CSG-Memo 284-2 MIT-LCS (1991).
15) Dally, W. J., Chien, A., Fiske, S., Horwat, W., Keen, J., Larivee, M., Lethin, R., Nuth, P., Wills, S., Carrick, P. and Fyler, G.: The J-Machine: a Fine-Grain Concurrent Computer, Proc. of IFIP World Computer Congress, pp. 1147-1153 (1989).
16) Dally, W. J., Chien, A., Fiske, S., Horwat, W., Keen, J., Nuth, P., Larivee, J. and Totty, B.: Message-Driven Processor Architecture: Version 11, MIT-AI memo No. 1069 (1988).
17) Kechler, S. W. and Dally, W. J.: Processor Coupling: Integrating Compile Time and Runtime Parallelism, to appear in Proc. of 19th ISCA (1992).
18) Agarwal, A., Chaiken, D., D'Souza, G., Johnson, K., Kranz, D., Kubiatowicz, J., Kurihara, K., Lim, B., Maa, G., Nussbaum, D., Parkin, M. and Yeung, D.: The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor, MIT/LCS/TM-454. b (1991).
19) Chaiken, D., Kubiatowicz, J. and Agarwal, A.: LimitLESS Directories: A Scalable Cache Coherence Scheme, Proc. of ASPLOS IV (1991).

(平成4年3月3日受付)



坂井 修一 (正会員)

昭和33年生．昭和56年東京大学理学部情報科学科卒業．昭和61年同大学院工学系研究科情報工学専門課程修了．工学博士．同年電子技術総合研究所入所．並列処理計算機，特に並列処理アーキテクチャ，相互結合網，スケジューリング問題などの研究に従事．平成3年4月より1年間米国 MIT 招聘研究員．情報処理学会研究賞（平成元年），同論文賞（平成2年度），元岡記念賞，日本 IBM 科学賞各受賞．現在，情報アーキテクチャ部計算機方式研究室主任研究官．